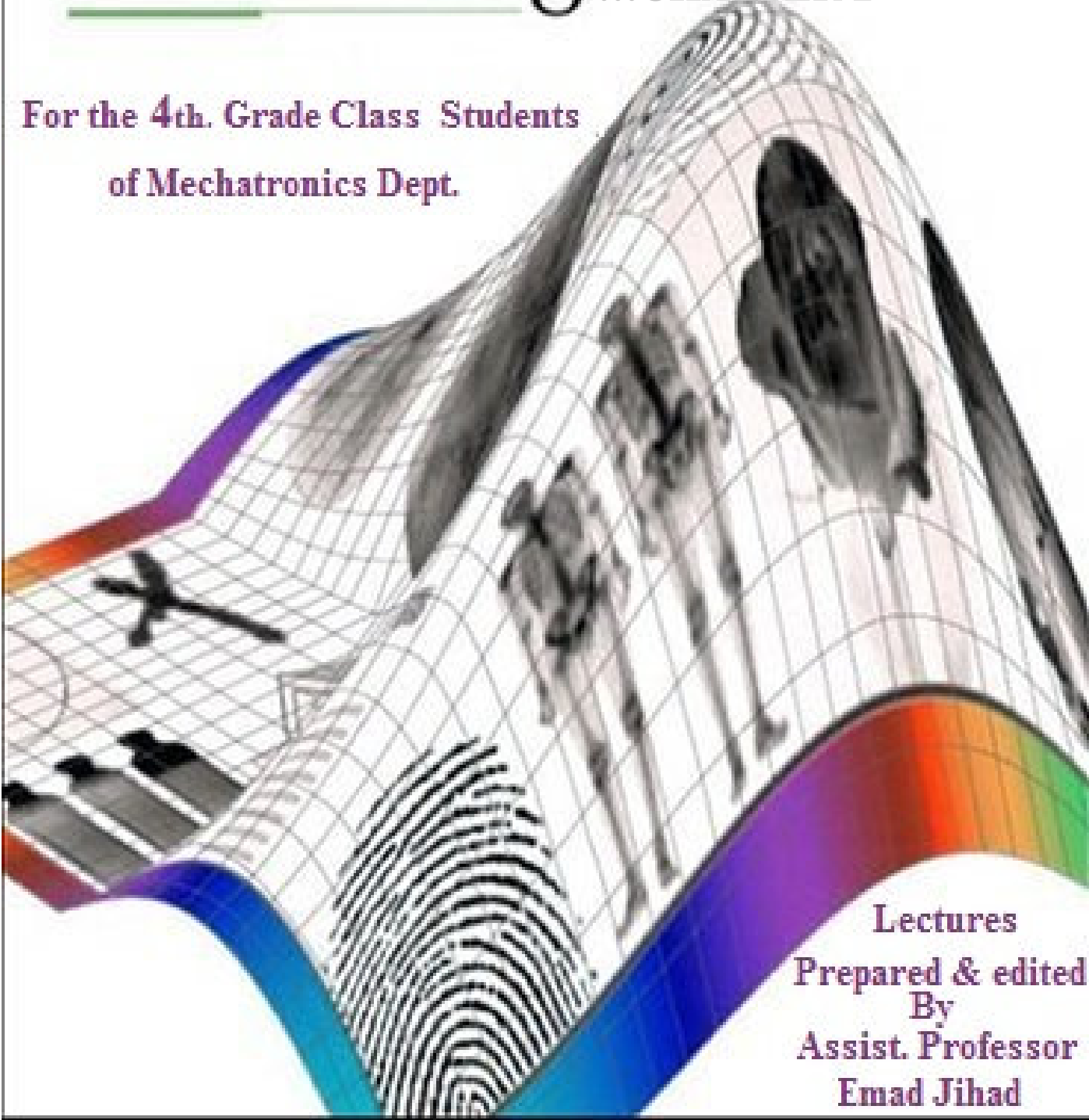


# Digital Image Processing

USING MATLAB®

## ...CHAPTER 2

For the 4th. Grade Class Students  
of Mechatronics Dept.









Lectures  
Prepared & edited  
By  
Assist. Professor  
Emad Jihad

## 2- Essentials of Image Processing in Matlab

There is several Software capable of processing digital images, some are programming languages such as VB, C++, Matlab etc, and others are Application software with certain ready tools to handle image processing.

**Matlab** is widely used in many Information Technology (IT), computer science and Other Technical computation Fields such as digital Image processing.

Images are held in files with different formats according to image characteristics. Matlab can deal with most of these formats, however, we will not use all of them, But only the most used and familiar formats (marked with red arrows in table below).

Format Name	Description	Recognized Extensions
BMP <sup>†</sup>	Windows Bitmap	.bmp 
CUR	Windows Cursor Resources	.cur
FITS <sup>†</sup>	Flexible Image Transport System	.fts, .fits
GIF	Graphics Interchange Format	.gif 
HDF	Hierarchical Data Format	.hdf
ICO <sup>†</sup>	Windows Icon Resources	.ico 
JPEG	Joint Photographic Experts Group	.jpg, .jpeg 
JPEG 2000 <sup>†</sup>	Joint Photographic Experts Group	.jp2, .jpf, .jpx, j2c, j2k
PBM	Portable Bitmap	.pbm
PGM	Portable Graymap	.pgm
PNG	Portable Network Graphics	.png 
PNM	Portable Any Map	.pnm
RAS	Sun Raster	.ras
TIFF	Tagged Image File Format	.tif, .tiff 
XWD	X Window Dump	.xwd

<sup>†</sup>Supported by imread, but not by imwrite

Fig (2-1): Image Formats table

### 2.1 Reading and Displaying image files

To read for example, a (.jpg) image format file, **imread** ('imagefilename.jpg') is to be used, then **figure** and **imshow** commands are used to show the image on figures window of Matlab.

Remember to put the image file format (extension name) along with the image file name used. Now, if we Do Not want to display numeric pixel element values of the image matrix, then we should put (;) after imread command, otherwise, matrix element values will be shown.

**Example 2-1-1:** Read an image ('MechaT.jpg') from current directory and display it.

```
>> imread ('MechaT.jpg');  
>> imshow ('MechaT.jpg')
```



But, if the image file was on another directory, then full Path must be used. For instance,

```
f = imread('D:\myimages\chestxray.jpg'); imshow (f)
```

Notice that an image array (matrix) name (f) is used to hold the data of the image and to avoid the repetition of full image file name.

In Matlab, **Whos** command can be helpful to give information about an image array.

**Example 2-1-2:** Display information about image file 'fruit-photo.jpg'?

```
>> f = imread('fruit-photo.jpg');  
>> imshow(f)  
>> whos f
```

Name	Size	Bytes	Class	Attributes
f	300x400x3	360000	uint8	



**Note:** There is an alternative of imread command using **load filename**, where filename is any image file name.

```
>> load fruit-photo.jpg  
>> imshow ('fruit-photo.jpg')
```

## 2.2 Writing image files

Images are written to the Current Directory using function **imwrite**, which has the following basic syntax:

imwrite (f, 'filename'). The filename must include a recognized file format extension.

**Example 2-2-1:** Write the image file in (Ex. 2-2) to the current directory with different name and format (.bmp).

```
>> imwrite (f, 'FP.bmp')
```

Here, a new file name is given with format bmp.

### 2.2.1 Additional imwrite parameters:

There are additional options or parameters used to write images, depending on the file format used.

For the (.jpg) format, there is the following function:

**imwrite (f, 'filename.jpg', 'quality', q)**

where **q** is an integer between 0 and 100 (the lower the number the higher the degradation due to JPEG compression).

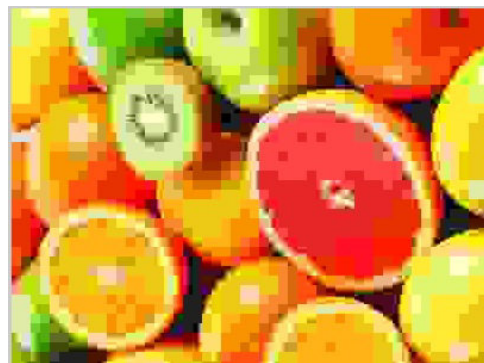
Now, if this parameter is not used, the resulting image will have the higher value of 100, otherwise, the resulting image will be less quality depending on q value used.

For a previous example (2-1-2), suppose we set q = 2, then we will get:

```
>> imwrite (f,'fcomp.jpg','quality',2)
>> c = imread ('fcomp.jpg');
>> imshow (c)
```

It's obvious the low quality of the new Written image.

Using **imfinfo** will give the full info about Both files as following:



```
>> imfinfo ('fruit-photo.jpg')
```

```
ans =

    Filename: 'fruit-photo.jpg'
    FileModDate: '11-17:40:01 2014-آب'
    FileSize: 76270
    Format: 'jpg'
    FormatVersion: ''
        Width: 400
        Height: 300
    BitDepth: 24
    ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
    CodingMethod: 'Huffman'
    CodingProcess: 'Sequential'
    Comment: {}
    Orientation: 1
    XResolution: 72
    YResolution: 72
    ResolutionUnit: 'Inch'
    Software: 'Adobe Photoshop CS2 Windows '
    DateTime: '2007:05:26 03:56:02 '
    YCbCrPositioning: 'Centered'
    DigitalCamera: [1x1 struct]
    ExifThumbnail: [1x1 struct]
```

```
>> imfinfo ('fcomp.jpg')
```

```
ans =

    Filename: 'fcomp.jpg'
    FileModDate: '07'13:26:47 2014-تشرين الأول-
    FileSize: 49956
    Format: 'jpg'
    FormatVersion: ''
        Width: 404
        Height: 301
    BitDepth: 24
    ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
    CodingMethod: 'Huffman'
    CodingProcess: 'Sequential'
    Comment{ } :
```

Now Checking between the two files and find That file size is less for the second one, and that some Information is missing too.

## 2.3 Image Classes and Types in Matlab

### 2.3.1 Classes

Although we work with integer coordinates, the values (intensities) of pixels are not restricted to be integers in MATLAB. The table (Fig 2.3.1) below lists the various *classes* supported by MATLAB and the Image Processing Toolbox† for representing pixel values. The first eight entries in the table are referred to as *numeric* classes.

The ninth entry is the *char* (character) class and, as shown, the last entry is the *logical* class.

Classes **uint8** and **logical** are used extensively in image processing, and they are the usual classes encountered when reading images from image file formats such as TIFF or JPEG. These classes use **1 byte** to represent each pixel.

Some scientific data sources, such as medical imagery, require more dynamic range than is provided by uint8, so the **uint16** and **int16** classes are used often for such data. These classes use **2 bytes** for each array element. The **floating-point** classes **double** and **single** are used for computationally intensive operations such as the **Fourier transform**.

Description	Name
Double-precision, floating-point numbers in the approximate range $\pm 10^{308}$ . (8 bytes per element).	<b>double</b>
Single-precision floating-point numbers with values in the approximate range $\pm 10^{38}$ (4 bytes per element).	<b>Single</b>
Unsigned 8-bit integers in the range [0, 255] (1 byte per element).	<b>Uint8</b>
Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).	Uint16
Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).	Uint32
Signed 8-bit integers in the range [-128, 127] (1 byte per element).	<b>Int8</b>
Signed 16-bit integers in the range [-32768, 32767] (2 bytes per element).	Int16
Signed 32-bit integers in the range [-2147483648, 2147483647] (4 bytes per element).	Int32
Characters (2 bytes per element).	Char
Values are 0 or 1 (1 byte per element).	<b>Logical</b>

Fig (2-3-1) Classes in Matlab

### 2.3.2 Image types in Matlab

As mentioned in Chapter 1(B/W, Gray level, Color RGB, and Color indexed).

#### 2.3.2.1 Generating Black and White (B/W) images

As mentioned before, B/W images are considered Binary or Logical pixel images with image matrix values of either 0 or 1 only.

The following Matlab program script shows how such images are generated.

**Example 2-3-2-1:** Write Matlab script that generates the following B/W images in a multiple figure cases: a Full Black pixels image, a Full White pixels image, reading and displaying a Monochrome image file, reading and display a small (4x14) logical image file and show its matrix elements, and finally Randomly generate a logical pixels matrix (Salt and pepper) image.

```
%*****
%* Generating B/W images      *
%* Program Name: ImageProEx01 *
%*****
clc
display ('Black and White image matrices Example!')
A = zeros(100,80);
B = ones (100,80);
C = imread('BW-image.bmp');
D = imread('BWbox.bmp');
E = int8(rand(100,80)); E = logical (E); % Define Logical elements
figure;

subplot(1,5,1)
imshow(A)
title ('Full (0) pixels Matrix')
whos A
subplot(1,5,2)
imshow(B)
title ('Full (1) pixels Matrix')
whos B
subplot(1,5,3)
imshow(C)
title ('bmp-MonoChrome image')
whos C
subplot(1,5,4)
imshow(D)
title ('B/W image')
whos D
subplot(1,5,5)
imshow(E)
title ('Logical Random pixels image')

display ('B/W Binary-Logical Image Matrix :')
D

whos E
```

(Notice that the size of logical images in Bytes is very small)

```

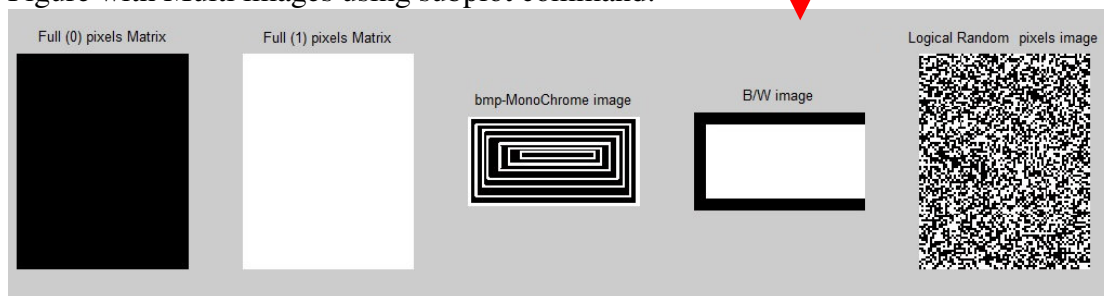
Command Window
Black and White image matrices Example!
Name      Size      Bytes  Class  Attributes
A         100x80      8000  logical
Name      Size      Bytes  Class  Attributes
B         100x80      8000  logical
Name      Size      Bytes  Class  Attributes
C         153x294     44982  logical
Name      Size      Bytes  Class  Attributes
D          8x14        112  logical

B/W Binary-Logical Image Matrix:
D =
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    1    1    1    1    1    1    1    1    1    1    1    1    1
    0    1    1    1    1    1    1    1    1    1    1    1    1    1
    0    1    1    1    1    1    1    1    1    1    1    1    1    1
    0    1    1    1    1    1    1    1    1    1    1    1    1    1
    0    1    1    1    1    1    1    1    1    1    1    1    1    1
    0    1    1    1    1    1    1    1    1    1    1    1    1    1
    0    0    0    0    0    0    0    0    0    0    0    0    0    0

Name      Size      Bytes  Class  Attributes
E         100x80      8000  logical
fx >>

```

Figure with Multi images using subplot command:





## 2.3.2.2 Generating Gray level images

Gray level images are generated either by software applications or by programming. In Matlab, Gray images can be generated either by Random way, or by converting color images to gray scale images as we will see later on.

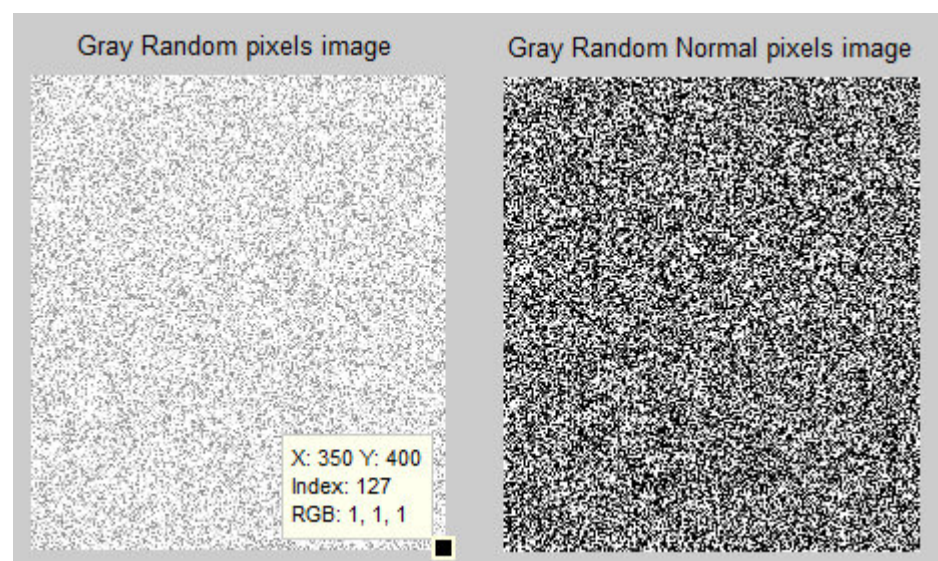
**Example 2-3-2-2-1:** Generate Randomly two Gray scale level images, the first one is Uniformly distributed and the second one is Normally distributed. The Program script is something like the following:

```
%*****
%* Generating Random Gray pixels image! *
%* Program Name: ImageProEx02          *
%*****
clc
display ('Gray level image matrices Example!')
display ('=====')
F = int8(rand (400,350)*255);% Creating Random Uniform Gray pixel
elements
E = int8(randn(400,350)*255);% Creating Random Normal  Gray pixel
elements
figure;
subplot (2,2,1)
imshow(F)
title ('Gray Random uniform pixels image')
subplot (2,2,2)
imshow(E)
title ('Gray Random Normal  pixels image')

whos F
whos E
```

After executing, we get:

```
Gray level image matrices Example!
=====
Name      Size      Bytes Class  Attributes
F   400x350    140000 int8
Name      Size      Bytes Class  Attributes
E   400x350    140000 int8
```





### 2.3.2.3 Generating Random image from Color image

**Example 2-3-2-3-1:** To scramble image color pixels, test the following Matlab script:

```
%*****  
%* Scrambling Color image Pixels randomly! *  
%* Program Name: ImageProEx04 *  
%*****  
clc  
I = imread('fruit-photo.jpg');  
imshow (I)  
size (I)  
r = randperm(300); % Generating random Permutation from 1~300  
c = randperm(400); % Generating random Permutation from 1~400  
J = I(r,c,:);  
figure  
imshow(J)  
title('Scrambled Image')  
xlabel('What is it?')  
display('Hit Return Key to Exit!');  
pause;  
close all
```



### 2.3.2.4 Reading and showing an indexed image file

Unlike other image types, an indexed image has a color index and a color map as we have seen in Chapter 1. Such kind of images is not read as other types, the following Matlab statements show an example of reading and showing such an image:

```
>> [b,bmap] = imread ('indi.tif');  
>> imshow (b,bmap)
```

Notice the use of index and color map as a matrix with (b) for example as index, and (bmap) as the colors map that both will have necessary pixels values. At this stage, we will not expand about this type of images!

## 2.4 Matlab image Tool

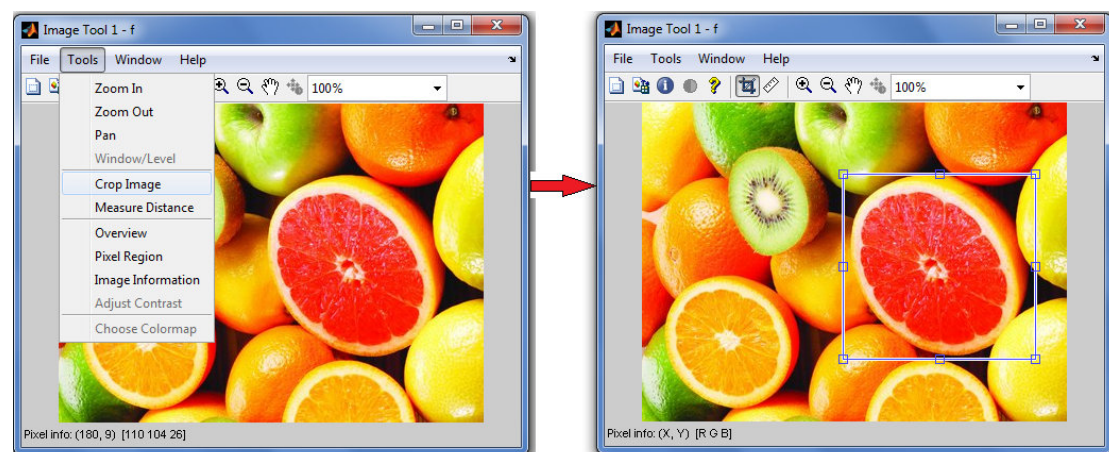
Image Tool of Matlab Image Processing Toolbox provides more interactive environment for images viewing and navigating.

Description	Tool
Displays information about the pixel under the mouse pointer.	Pixel Information
Superimposes pixel values on a zoomed-in pixel view.	Pixel Region
Measures the distance between two pixels.	Distance
Displays information about images and image files.	Image Information
Adjusts the contrast of the displayed image.	Adjust Contrast
Defines a crop region and crops the image.	<b>Crop Image</b>
Shows the display range of the image data.	Display Range
Shows the currently visible image.	Overview

To start the Image Tool, use the **imtool** function. For example, the following statements read an image from a file and then display it using imtool:

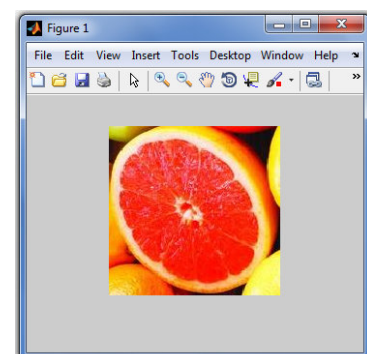
**Example 2-4-1:** After using image tool, use from Tools "Crop Image" to cut an area required for example and save it to a new .jpg file.

```
>> f = imread('fruit-photo.jpg');
>> imtool(f)
```



After determining the required area, use **Save as** from File Command on Image Tool. Now, read and show the cropped image as follows:

```
>> cf = imread('Orange.jpg');
>> imshow(cf)
```



**Example 2-4-2:** Display pixel information while the mouse is moving over the image

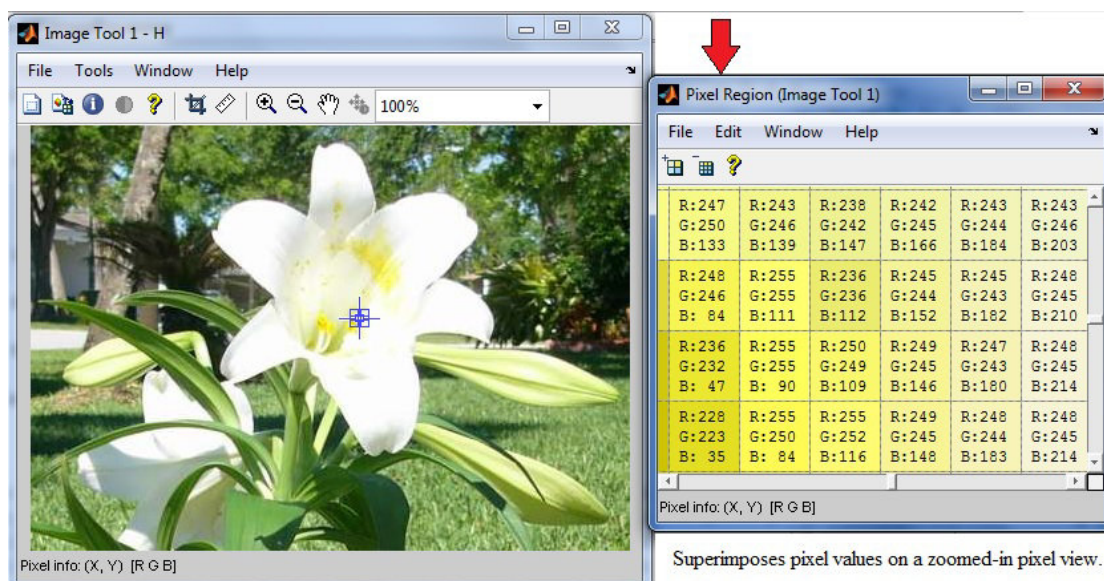
```
>> H = imread('BasHot.jpg');  
>> imtool (H)
```



Pixel info includes cursor pointer coordinates and intensity value (RGB).

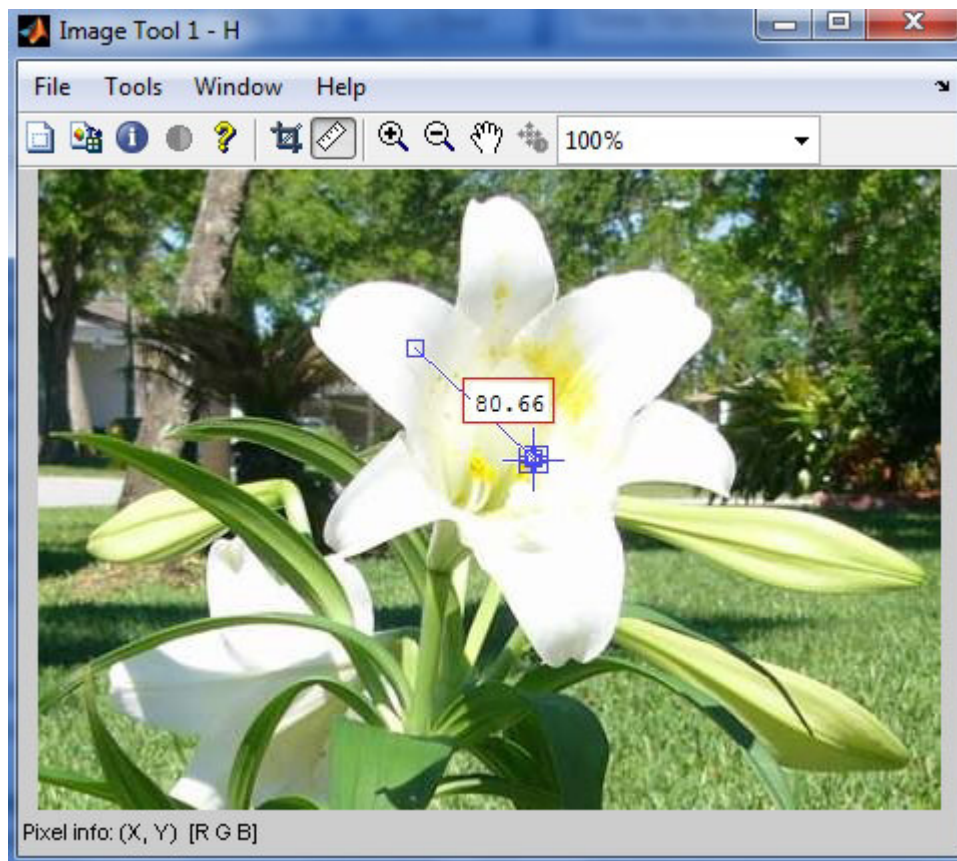
**Example 2-4-3:** Use Pixel Region tool to show the RGB values under:

```
>> H = imread ('flower-w.jpg');  
>> imtool (H)
```





**Example 2-4-4:** Use Measure Distance tool to find the distance between two pixels:



**Exercise:** Try to find out other tools available!

## 2.5 Converting between image types



It is important some times to convert an image type to another, in Matlab this is done by using special conversion functions that some are illustrated in Fig. (2.5.1) below:

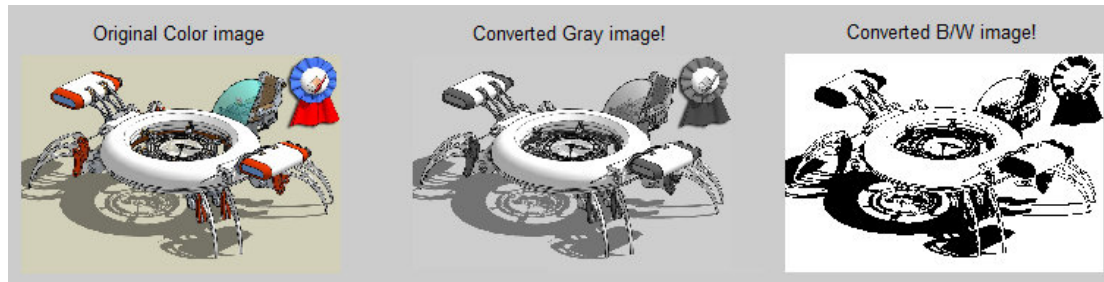
Function	Use	Format
ind2gray	Indexed to Greyscale	<code>y=ind2gray(x,map);</code>
gray2ind	Greyscale to indexed	<code>[y,map]=gray2ind(x);</code>
rgb2gray	RGB to greyscale	<code>y=rgb2gray(x);</code>
rgb2ind	RGB to indexed	<code>[y,map]=rgb2ind;</code>
ind2rgb	Indexed to RGB	<code>y=ind2rgb(x,map);</code>

**Fig (2.5.1)**

Where y in the table's format field is an image file matrix that been read by imread.

Note: We can also convert Gray level image to B/W by using: **im2bw** function.

**Example 2-5-1:** The following script is showing an example about image type conversions: Suppose we have one RGB color image and want to convert it to Gray level then to B/W logical image as the following:



Also show a comparison of these types information at the end.

The Matlab program script could be something like this below:

```
%*****
% Converting Color RGB image to Gray scale image! *
% Program Name: ImageProEx5 *
%*****
clc
close all
y = imread ('TMach.jpg');
x = rgb2gray (y);
w = im2bw(x);
subplot (3,3,1)
imshow (y)
title ('Original Color image')
subplot (3,3,2)
imshow (x)
title ('Converted Gray image!')
subplot (3,3,3)
imshow (w)
title ('Converted B/W image!')
display ('Checking the info of each type:')
display ('=====')
whos y
whos x
whos w
```